Gamifier: An Ensemble Program for the Game Design Process

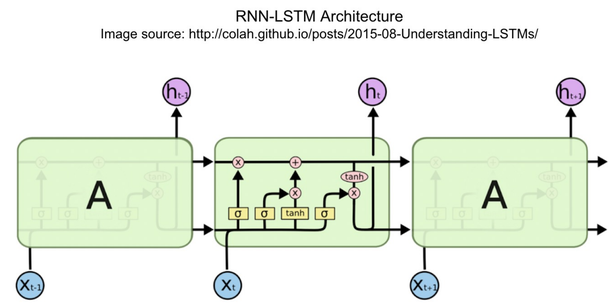
Eleanor Ye (jye20) | Matthew Stephens (mstephe7)
Brown University, CS1470: Deep Learning

Introduction

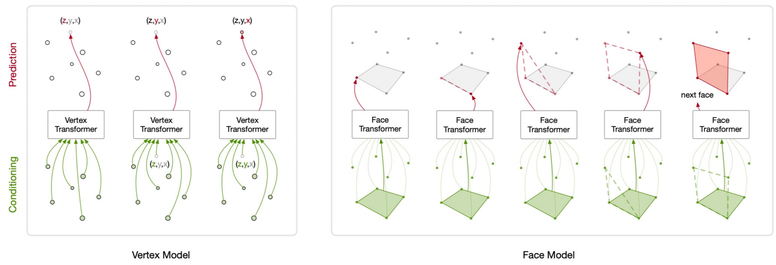
We present Gamifier: an ensemble program consisting of three deep learning models designed to simplify the game design process. The first of the models is a RNN language model named “Thinker” — trained on both Wikipedia articles and two Harry Potter books — that takes in a user inputs (say, a single idea for an object to be added to the game) and outputs a list of ideas that are related to the first idea. Once this list has been created, it is fed into DeepMind’s PolyGen model — an “Autoregressive Generative Model of 3D Meshes” — which generates 3D objects as meshes from text descriptions (DeepMind, Polygen). Additionally, the list of ideas is inputted into a Conditional Generative Adversarial Network (cGAN) — rebuilt following inspiration from the network “Coloring with Words: Guiding Image Colorization Through Text-based Palette Generation” of Bahng et al. (Bahng et al., Coloring with Words) — in order to output a set of color recommendations for the objects outputted by the initial ThinkerRNN model. Through combining these three models, we nicknamed the new umbrella model “Gamifier”, as we believe it will simplify the game design process and unlock a range of potential applications in 3D scene creation, augmented reality, and virtual reality.

Model Architectures

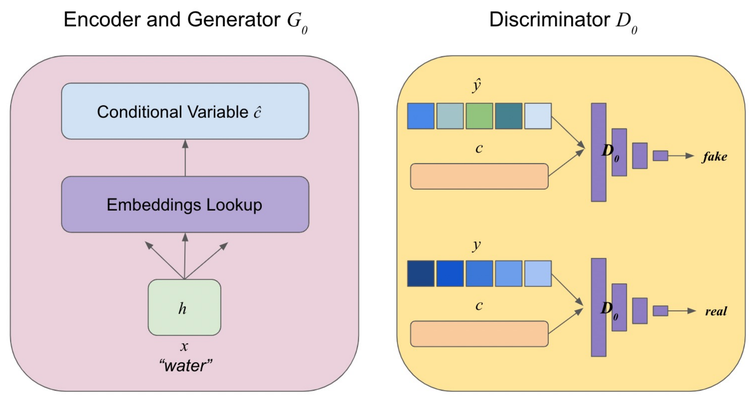
ThinkerRNN (LSTM Architecture)



PolyGen



PaletteGAN



Sources

PolyGen

- Paper: *PolyGen: An Autoregressive Generative Model of 3D Meshes*, Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, Peter W. Battaglia
- GitHub: https://github.com/deepmind/deepmind-research/tree/master/polygen

Text-to-Palette Generation Network

- Paper: Coloring with Words: Guiding Image Colorization through Text-based Palette Generation
- GitHub: https://github.com/awesome-davian/Text2Colors

Moving Forward

- Reconstruct scenes for the game designer given a text description of the scene.
- Apply the PaletteGAN color palettes to the generated meshes themselves (for instance, using Blender)
- Re-implementing other 3D reconstruction networks, such as the work of Yufie Ye et al. in *Shelf-Supervised Mesh Prediction in the Wild*, where a neural network is trained to produce its own 3D reconstructions from image input.

Challenges

Workflow

- PolyGen model has a dataset of only 55 objects (129 words), whereas the RNN is trained on a corpus of thousands of words (two Harry Potter novels and hundreds of Wikipedia articles), and the PAT dataset contains 4312 unique words. Hence, PolyGen’s list of available words places limitations on the other two models.

ThinkerRNN

- Finding the cosine similarity between embeddings of words holding little-to-no similarity to each other.

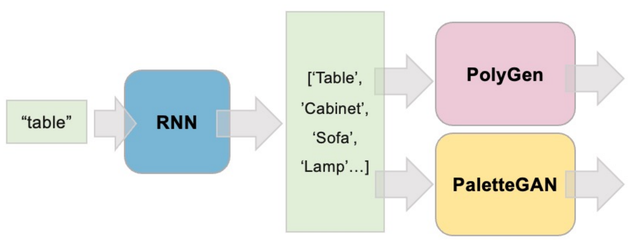
PolyGen

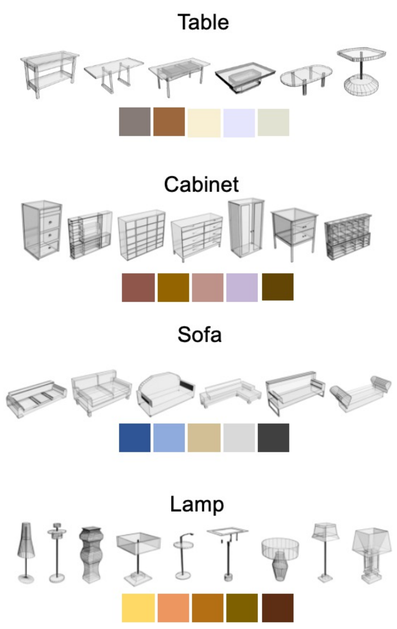
- The PolyGen model cannot easily be reimplemented in pytorch; this owes mainly to PolyGen’s heavy dependance on the underlying sonnet library (used throughout the model in inheriting classes, functions, abstractions and function decorators).
- Downloading the ShapeNet Core V2 dataset (~25GB), and thereafter retraining the model on the dataset.

PaletteGAN

- Unable to use 300-dimensional GloVe word embeddings.
- Unable to load .pth files within our tensorflow reimplementation.

Workflow





Results

Quantitative Results

ThinkerRNN:
ThinkerRNN Final Perplexity Score: 50.860847
PaletteGAN:
Epoch 0 generator loss: 92.9266 , discriminator loss: 0.010133302
Epoch 500 generator loss: 50.63282 , discriminator loss: 1.6480e-05
Epoch 1000 generator loss: 44.45528 , discriminator loss: 1.2741e-05

Example Qualitative Results

ThinkerRNN:
Input: Lounge, Output: `[('lamp', 2.7432356), ('remote_control', 2.8682358), ('video_display', 2.9316287), ('headphone', 3.22737)]`
PolyGen and PaletteGAN: